图文 077、动手实验：自己模拟出JVM栈内存溢出的场景体验一下！

详情 评论

动手实验：
自己模拟出JVM栈内存溢出的场景体验一下！

狸猫技术窝专栏上新，基于**真实订单系统**的消息中间件（mq）实战，重磅推荐：

未来3个月，我的好朋友原子弹大侠将带你一起，全程实战，360度死磕MQ

（点击下方蓝字进行试听）

从 0 开始带你成为消息中间件实战高手

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情在评论区留言提问，我都会逐一答疑

（**ps**：评论区还精选了一些小伙伴对**专栏每日思考题的作答**，有的答案真的非常好！大家可以通过看别人的思路，启发一下自己，从而加深理解）

如何加群：购买了狸猫技术窝专栏的小伙伴都可以加入**狸猫技术交流群**。

（群里有不少**一二线互联网大厂的助教**，大家可以一起讨论交流各种技术）

具体**加群方式**请参见文末。

（**注**：以前通过其他专栏加过群的同学就不要重复加了）

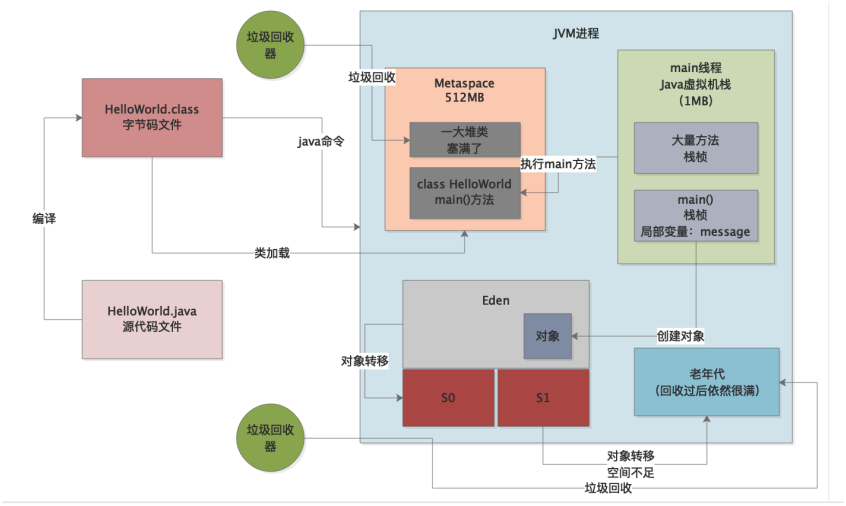
1、前文回顾

上文我们已经给大家演示了如何通过在系统中动态生成大量的类来塞满Metaspace区域，进而引发Metaspace区域的内存溢出。

这篇文章我们就来带着大家通过代码来示范一下栈内存溢出的场景。

2、重新分析一下JVM中的栈内存

咱们先来简单的回顾一下JVM的整体运行原理，大家不要忘记下面这张图，必须牢牢印刻在自己的脑海里。



既然大家已经对Metaspace这块区域的内存溢出理解的很深刻了，那么接着我们来回顾一下栈内存这块区域的内存溢出。

讲到这里我们先带着大家来思考一下，JVM进程到底会占用机器上多少内存?

先不考虑一些细小的其他内存区域，就仅仅考虑一下最核心的几块就可以了，包括了：Metaspace区域，堆内存区域，各个线程的栈内存区域。

Metaspace区域我们一般会设置为512MB左右的大小，这个大小只要你代码里没有自己胡乱生成类，一般都是绝对足够存放你一个系统运行时需要的类的。

堆内存大小，之前在分析GC的时候给大家大量的分析过，堆内存一般分配在机器内存的一半就差不多了，毕竟还要考虑其他人对内存的使用。

那么最后一块内存区域我们之前一直没怎么给大家说过，就是栈内存区域。

其实大家考虑一个最基本的线上机器配置，比如4核8G的线上机器，其中512MB给了Metaspace，4G给了堆内存（其中包括了年轻代和老年代），剩余就只有3G左右内存了，要考虑到操作系统自己也会用掉一些内存。

那么剩余你就认为有一两个GB的内存可以留给栈内存好了。

通常来说，我们会设置每个线程的栈内存就是1MB，假设你一个JVM进程内包括他自带的后台线程，你依赖的第三方组件的后台线程，加上你的核心工作线程（比如说你部署在Tomcat中，那就是Tomcat的工作线程），还有你自己可能额外创建的一些线程，可能你一共JVM中有1000个线程。

那么1000个线程就需要1GB的栈内存空间，每个线程有1MB的空间。

所以基本上这套内存模型是比较合理的，其实一般来说，4核8G机器上运行的JVM进程，比如一个Tomcat吧，他内部所有的线程数量加起来在几百个是比较合理的，也就占据几百MB的内存，线程太多了，4核CPU负载也会过高，也并不好。

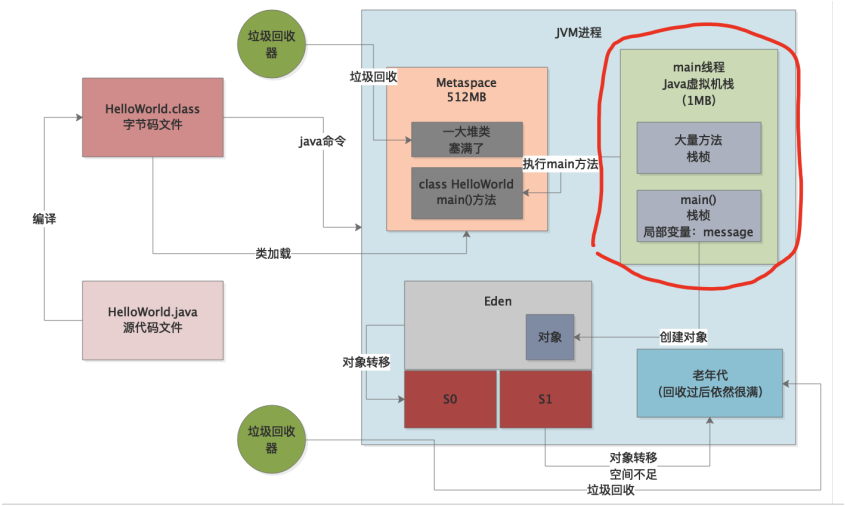
所以Metaspace区域+堆内存+几百个线程的栈内存，就是JVM一共对机器上的内存资源的一个消耗。

所以大家这里也能理解一个道理，你要是给每个线程的栈内存分配过大的空间，那么会导致机器上能创建的线上数量变少，要是给每个线程的栈内存相对较小，能创建的线程就会比较多一些。

当然一般来说，现在都建议给栈内存在1MB就可以了。

3、回顾一下栈内存溢出的原理

接着我们来回顾一下栈内存溢出的原理，其实特别的简单，大家看下图中画红圈的地方，其实每个线程的栈内存是固定的，要是你一个线程不停的无限制的调用方法，每次方法调用都会有一个栈桢入栈，此时就会导致线程的栈内存被消耗殆尽。



但是通常而言你的线程不至于连续调用几千次甚至几万次方法，对不对？一般发生这种情况，只有一个原因，就是你的代码有bug，出现了死循环调用，或者是无限制的递归调用，最后连续调用几万次之后，栈内存就溢出了，没法放入更多的方法栈桢了。

4、用一段代码示范一下栈内存溢出

下面大家先看一段代码：

```
public class Demo2 {
    public static long counter = 0;
    public static void main(String[] args) {
        work();
    }
    public static void work() {
        System.out.println("目前是第" + (++counter) + "次调用方法");
        work();
    }
}
```

上面的代码非常简单，就是work()方法调用自己，进入一个无限制的递归调用，陷入死循环，也就是说在main线程的栈中，会不停的压入work()方法调用的栈桢，直到1MB的内存空间耗尽。

另外大家需要设置这个程序的JVM参数如下：-XX:ThreadStackSize=1m，通过这个参数设置JVM的栈内存为1MB。

接着大家运行这段代码，会看到如下所示的打印输出：

目前是第5675次调用方法
java.lang.StackOverflowError

也就是说，当这个线程调用了5675次方法之后，他的栈里压入了5675个栈桢，最终把1MB的栈内存给塞满了，引发了栈内存的溢出。大家看到StackOverflowError，就知道是线程栈内存溢出了。

5、本文总结

本文带着大家用代码实验了一下栈内存的溢出，大家可以看到1MB的栈内存可以让你连续调用5000次以上的方法，其实这个数量已经很多了，除了递归方法以外，一般根本不可能出现方法连续调用几千次的。

所以大家就知道，一般这种栈内存溢出是极少在生产环境出现的，即使有，一般都是代码中的bug导致的，本周后续会用真实的生产案例告诉大家当时我们线上系统是什么情况下出现过偶发性的这种问题。

End

如何加群?

添加微信号：Lvgu0715_ （微信名：绿小九），狸猫技术窝管理员

发送 Jvm专栏的购买截图

由于是人工操作，发送截图后请耐心等待被拉群

最后再次提醒：通过其他专栏加过群的同学，就不要重复加了

狸猫技术窝其他精品专栏推荐：

21天互联网java进阶面试训练营（分布式篇）